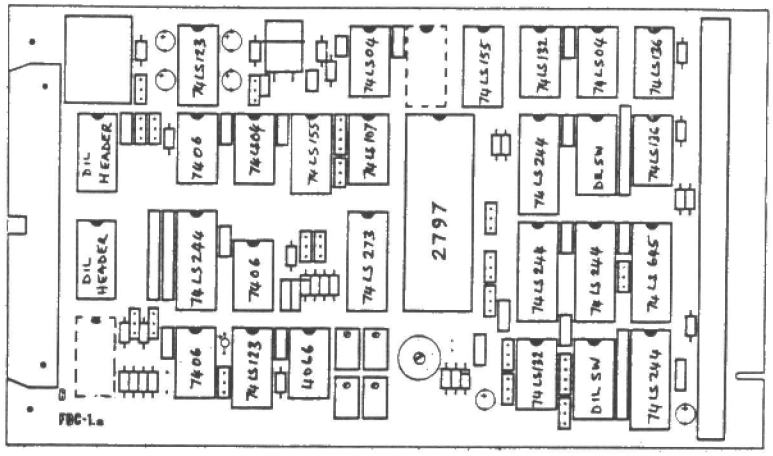
Document Ref: FDC1-BDS
Issue No: 1. July 1986

Greenbank Electronics
Telephone: 051-645 3391

Interak 1

DISK INTERFACE

Floppy Disk Interface



FDC-1 Floppy Disk Interface Card

FDC-1 FEATURES

* Uses Industry Standard 2797 FDC "Single Chip" Controller IC.
* 2797 is Compatible with earlier Multiple Chip 1797 Controller.
* Single Density or Double Density Operation.
* 40 or 80 Track on Small Size Diskettes ("Quad" density).
* True Double Sided Operation with twin headed Drives (2797 has Side Select Output).
* With Z80A at 4 MHz or faster, without wait states, allows data transfer rates at up to 500 Kilobits per second (twice the maximum for some competing systems).
* Above rate achieved without need for "suspect" design techniques like making the system "hang", activating the wait line for synchronisation etc.
* Controller Chip Type 2797 has phase locked loop data separator and write precompensation logic for added data integrity.
* CRC (Cyclic Redundancy Check) Bytes Automatically Generated and Confirmed.
* Track to track stepping rates are programmable, allowing drives with different step rate to be used.
* Fast 3ms Stepping Rate possible for Small Size Drives (Quieter Operation).
* 2797 Commands Include Multiple Sector (eg All Sectors) Read/Write in single revolution of Diskette, With CRC Error Checking.
* 2797 Commands Include Programmable Control of Track to Track Access, and Head Load Timing.
* 2797 Commands include Automatic Seek With Verify.
* Whole Track Read Command allows Diagnostic Investigation, and "Cracking" of Unknown Disk Formats.
* Programmable Sector Length, 128, 256, 512, 1024 Bytes/Sector.
* Write Protect Input prevents accidental corruption of Data on Diskette.
* Supports IBM 3740 FM (Frequency Modulated, ie Single Density) Format.
* Supports IBM System 34 MFM (Modified Frequency Modulated, ie Double Density) Format.
* On board 4.0 MHz Crystal Oscillator makes operation independent of System CPU Clock.
* All Signals to and from 2797 Controller Chip are Buffered; ie No Drive or Computer Interface Signal is Connected Directly to any pin of the 2797.

* Board can be used with up to 4 standard 8 inch, 5.25 inch, 3.5 inch or 3 inch disk drives.
* Controller can be switched from 8 inch to smaller sizes purely under software control, allowing mixed drives to be used.
* Plug in DIL Headers allow choice of 50-way or 34-Way Shugart Compatible Interface.
* Artificial ready signal allows use of drives which do not have a true ready signal.
* Head Load Time Jumper Selectable.
* "Motors On" Monostable keeps motor running for a short period after drive has been deselected so that there is no delay to allow motors to reach speed every time a drive is selected.
* If the drive does not have motor control, links are provided to disable this feature.
* SIL Pack Termination Resistors pluggable for alternative termination requirements.
* I/O Port Addresses are selectable by DIL Switch.
* Individual Adjustments Available for Write Precompensation for both 8" and smaller drives. (Other Designs fudge this, providing only one setting for the two different types.)
* CMOS Quad Analogue Switch Connects appropriate Phase Locked Loop components to suit drive in use, either 8" or smaller. (Other designers fudge this, using the same components for all sizes.)
* Individual Adjustments Available for Read Pulse Width for both 8" and smaller diameter drives.
* Single Adjustment for Voltage Controlled Oscillator suits all sizes of diskette.
* Numerous Jumper Link Selections for Individual User Preferences.
* Provision for Software Selection of Track for Write Enable Precompensation.
* Setting up Service available.
* Works with 4 MHz or faster Z80A, with no additional "Wait" States.
* Board is supported by DMON boot/monitor program and Interak implementation of CP/M.
* Epoxy Glass, Tinned.
* Designed specifically for Interak, but suitable also for other buses e.g. Kemitron KBUS-5 and KBUS-12.
* Double Sided, plated through hole construction.
* First supplied in 1985.
* International Size Card (4.5" x 8").
* Provision for Mounting of Metal Card Front.
* Gold-Plated 0.1" Pitch Edge Connector on both "A" and "B" sides.
* Green Solder Resist Screen on both "A" and "B" Sides.
* 18-pin 0.3"/0.6" DIL Patch Area Provided for User's Own Purposes.
* Needs Only a Single Rail 5V Supply.
* Calculated Supply Current mA typical, mA max.
* Buffered where necessary to reduce Bus Loading to 1 "LS" Load per Line.
* Our Name is not shown on the Card, so it is Ideal for OEM Use.

FDC-1 DESCRIPTION

The disk interface board for Interak is called "FDC-1", and allows the use of all sizes and speeds of disk drives, 3", 3.5", 5.25", 8"; 300 rpm, 360 rpm and 600 rpm; data transfer rates 125 kbit/s, 250 kbit/s, and 500 kbit/s; single, double and so-called "quad" density, single or double sided, with 1, 2, 3, or 4 drives per system (more can be added, either by modifying the FDC-1 card, or by adding further FDC-1 Cards).

Installation is very straightforward: the card simply plugs into any slot of the standard ISBUS backboard, and connects to the disk drives with no more difficulty than connecting the computer to a cassette tape recorder. According to the type of drives a single 34 or 50-way ribbon cable is used to carry the interface signals, "daisy-chained" to all of the disk drives. 1, 2, 3, or 4 drives can be connected on one interface cable. Standard signal allocations have been used (where standards exist), so that in the majority of cases the drives are simply plugged straight in without further ado; but an excellent feature of the FDC-1 design is that extensive use has been made of plug-in headers and jumper links so that a much wider range than normal of drive types can be accommodated as simply as possible, without fundamental changes to the tracks and circuitry on the board.

A particularly outstanding feature of the FDC-1 card is that the fundamental electrical changes to suit different types of disk drives are carried out in software rather than by jumper links and the like. For example I am typing this on a computer which has a mixed collection of 8" single sided drives, and 3.5" double sided drives, and the transfer of data from one disk to another takes place as easily as between disks of identical size.

Impish

The design of a floppy disk interface is a task which sorts the men from the boys. All aspects of computer design demand that the hardware designer understands the software aspects, and vice versa, and nowhere is this more pronounced than in the design of a floppy disk interface. It is a task which is at first sight deceptively simple, and it is indeed quite easy to design a circuit which will interface one type of disk drive to one particular computer. But the design cannot stop there. To be of general use, the interface must cope adequately with all combinations of different disk drives under all conditions, and with an eye to the future. I always gain some impish enjoyment when I see big, very well known names dig a pit for themselves and fall right in. I can see something has gone wrong in someone's design when I read in a computer magazine review that for example "Canon disk drives are very noisy when seeking a track", although my personal experience of that model drive is that in fact they are particularly quiet. I know what the problem is in that case, because I fell for it in the initial design stages of the FDC-1 card. It is simply a matter of stepping

the head of the disk drive too slowly. In this example the Canon Disk drive required an exemplary 3ms stepping rate from track to track, and the computer which was the subject of the magazine review used a disk interface design which only allowed a minimum stepping rate of 6ms. The heads in disk drives are moved by a stepper motor, pulsed magnetically one step at a time, and they are very quiet at the correct speed but very noisy if run slower (the effect is something like a pneumatic drill as every 6ms the motor is brought to rest and then accelerated up to maximum speed again, instead of running smoothly at 3ms).

Other examples, in similar vein, are when one manufacturer's disk drives are unjustly accused of being unreliable for certain well known educational computers, when in fact they are fine when used in a proper design. The fault is a bit more difficult to explain, but in this case is traced to the design of the computer (notwithstanding it has sold in its hundreds and thousands, and has a circuit board which has been through 10 or more redesigns). Simply, it is a matter of starting to read or write a track before some mechanical part of the drive has stabilised, the motor speed for example, or the magnetic heads themselves. The mystery to me is not so much why some makes of drives don't work, but how there are any makes which do, and indeed how did a firm with such a reputation drop such almighty clangers?

I have mentioned these points, not so much for the fun of "knocking" competitors, which, although fun, does not win me many friends, but to show what a delicate art Floppy Disk Drive Interface Design is.

Refinement

I would like to acknowledge the great help given to me by the talented users, Bob Eldridge and the late Wolf Schroeder, in testing and refining the design. The design was done quite quickly some years ago, but took a further two years of testing and refinement to cover all these subtle points I mention.

The angels are on my side in this design, because I took a chance and chose the very best chip for the purpose, the Western Digital 2797 even though it was at the time impossibly expensive, about 70 pounds or so. I was gambling that such a chip would prove so good that everyone would make it and the price would fall. The angels have been of help here because the chip is now second sourced by Texas Instruments, Siemens, and probably others, and is under 20 pounds, which is worth every penny for the work it does. (You may know of cheaper chips, but I think you will find that they don't work alone, and when the price of the related support chips is added they're not such a bargain.) A "single chip" controller such as the 2797 has the great benefit from a constructor's point of view that all the clever stuff goes on inside and no particular skill is needed of the user to achieve a perfect system. (Some setting up adjustments are needed, but if the user cannot make them himself, we offer a setting up service.)

A feature of the 2797 design is that it is similar in use to Western Digital's earlier multiple chip controllers, 1797, etc, even the original 1771 single density controller has some aspects in common. There are nowadays other offerings from Western Digital for floppy disk controllers, such as the 1770 series. On the face of it these look even better, because, again, they are compatible with all the others from a software point of view, and better still need no setting up whatever. Why then have I rejected the 1770 series in this design? The first reason is that the maximum data transfer rate of the 1770 is 250 Kilobits per second, half that of the 500 Kilobits per second 2797. "So what?" you might say, minifloppies (ie 5.25") and smaller diskettes only transfer data at 250 Kilobits per second anyway. 8" drive rates are 500 Kilobits per second, but who wants 8" disks? But consider: there are small drives which use the higher rate, for example the Sony 3.5" drives which rotate at 600 revs per minute instead of the normal 300, and the IBM AT 5.25" drives which have the same 1.6 Megabyte capacity and data transfer rate of 8" drives.

Pants Down

There is talk of some manufacturers putting 2 Megabytes or more of storage on a 3.5" floppy disk - if the rotational speed of the disk remains the same the data rate will double. If this happens users of the 1770 series chips will be caught with their pants down again. I well remember a debate with an earnest but misguided user some years ago who was trying to persuade me to use the Intel 8271 single density disk controller because (he said) it was superior to Western Digital designs, and (he said) single sided single density storage was enough for anyone, after all who would ever want more than 100K of storage per disk! The widespread use of the 8271 turned out to be a disaster, but I shan't go on, because I don't want to start "knocking" again!

Suffice it to say that I like the 2797 because it does have that reserve performance of 500 Kilobits per second built in. The way this doubling is achieved is by doubling the clock input frequency to the chip; virtually everything goes twice as fast, and enables us to offer that elusive 3ms (instead of the previous maximum 6ms) track to track rate for the head stepping motor.

Million Pounds

It is easy to be casual about the data transfer rate of 500 Kilobits per second, but this part of the design is a part which really made my brain hurt. Remember that once a disk read or write operation begins it must continue for completion; there is no possibility whatever for stopping the diskette's rotation for a moment to allow the system time to catch up, Felix just keeps on walking. The 2797 chip works valiantly to output the data and calculate the cyclic redundancy checks bytes etc, at this maximum rate if required (clever chip!), but it is the system designer's responsibility (ie mine; the buck stops here) to ensure that the 2797 is never kept waiting for data. If the time comes for data to be output and there aren't any, the 2797 has to write blanks and note the error. (This is the way electric light bills for a million pounds per quarter are issued to unsuspecting consumers I suppose.)

The most stringent timing occurs when writing to the disk. You will see from the 2797 data sheet (see the end of this document for the price of the data sheet) that although the terminology is quaintly reminiscent of an English garden party, (the parameters are called "T set" and "t service", sadly no Q cumber sandwiches), the pace is fast and furious. "t service" is the killer; this is the maximum time the 2797 can wait after requesting a fresh byte of data before the system provides it. The worst case time is 11.5 microseconds. It may sound like a long time to you but don't forget it takes a million of these just to make one second. The Z80A is pretty fast and can execute some of its simplest instructions in a one millionth of a second, but some take longer and the 11.5 microsecond limit to do everything it has to do is pretty severe. Just for the record (excuse the pun) what has to be done is the DRQ (data request) signal from the 2797 first has to be examined because this is how the 2797 requests fresh data to write on the disk, then the data has to be supplied, and the DRQ signal monitored again; this must carry on for as many bytes are to be written, 128 for the smallest sectors to several thousand for a whole track write. Concurrently another 2797 signal, INTRQ (interrupt request, to interrupt the process at the end), must also be monitored, and if detected the sequence concluded.

Coward's Way

All this has to be done in just a few lines of code, so that it does not exceed the worst case "t service" time of 11.5 microseconds. The use of interrupts is often thought to be pretty fast and ideal for this sort of purpose, but unless some naughty chances are taken even a Z80A at 4 MHz running without wait states, as we do, can't cope using this method. At first sight the task looks impossible (the problem is when DRQ is not found and a loop has to be executed to come back and look again; in the worst case DRQ will have been valid but undetected for most of that loop and valuable

microseconds will have been ticking away) and so it seemed to me at first. The coward's way out is to alter the specification to say small size disks can be read in single and double density, but 8" single density only. He can get away with this because most people don't care about 8" double density ... until they need that rate. (The 8" double density data rate is used on high performance versions of smaller diameter drive, and who is to say they won't catch on, the IBM-AT personal computer for example uses the 8" data rate on its high capacity 5.25" drives.)

Cold Fish

Fortunately the coward's way out did not have to be mine, because help arrived in the form of a knight in shining armour: a good old Interak user from way back, Bob Eldridge. (It's highly embarrassing when the users become cleverer than the system designers, but there it is, credit where credit is due, he saved the day.) The trick is to place DRQ and INTRQ on particular bits of a particular port (you can buy the FDC-1 Manual for the precise circuit, you can't have this secret for free!) in such a way that the state of both signals can be tested in a single instruction. The rich and colourful Z80 instruction set which looks so haphazard to some snooty cold fish in the computer industry ("It's so unstructured my dear!") nevertheless contains some jewels. One such is a special form of the INPUT instruction which simultaneously inputs data and alters internal flags depending on particular patterns of data. Using this combination of a clever hardware arrangement for DRQ and INTRQ, and pulling a wonderful Z80 instruction code out of the bag, the job is done. (This incidentally is why I still like the Z80A, and closely related chips; the Z80A was designed by real men for real men to use, later chips were designed by computers for computers to use, shiny new and chrome plated they may be, but they can't pull you out of trouble when you're stuck in the mud!)

Cataleptic

A method I considered when I was desperate, and which I discarded with Bob Eldridge's help (thanks Bob!) in favour of the wonderfully elegant solution I use today, has been used by some others in other designs (assuming they can cope with 8" double density that is). I shall enjoy offering a few words in criticism of the technique, because they come from designers who obviously think they're much cleverer than the likes of me! There are variations on the theme but in essence the method is to avoid the nasty long loop I mentioned when DRQ is found not valid and the routine has to go round and look again, wasting time it can ill afford in the process. The method is on the face of it quite appealing. The software is written assuming that DRQ is always ready (although it isn't of course) but whenever DRQ (and INTRQ) are tested, the computer "hangs", perhaps via activation of the "wait" line. It remains in this cataleptic state like Rip van Winkle until it is awakened by a kiss from a prince called DRQ who later turns out to be a frog (or maybe I've got my fairy stories mixed up, I said I don't favour this method.)

The problem with letting a computer "hang" in this way is it may have to wait for a hundred years for the frog DRQ to turn up, even if he ever does. I feel that allowing the system to "hang" waiting for something to happen which is outside its control is to be avoided. (Of course not everyone has Bob Eldridge helping them, so they have to make their way as best as they can.) If the system hangs for any significant length of time disasters can befall: dynamic RAM refresh, if controlled by the CPU, may fail, unknown to the user if the delay is not too long, but some of his data can be ruined; interrupts (if used) can pile up and be ignored, or in an extreme case the system must be reset entirely and begin again at square one.

Obviously the above catalogue of disaster doesn't always occur with a shaky design, but it is harder to give protection against some oversight on the user's part. For instance a system which "hangs" catastrophically when say a diskette is accidentally inserted upside down (mercifully, this is impossible with 3.5" drives) is not so much fun to work with as one which doesn't. Who wants a write to proceed after a delay which might have junked all the data to have been written. "Not I", said the fly.

Needs Ignored

The other point I mentioned is one which does need some careful consideration, and that is the fact that the 2797 needs setting up. Yes, this is unpleasant but if these parameters were preset how are we to allow the user to trim such items as the "read pulse width" and the "write precompensation" width? The inferior chips to the 2797 usually give a fixed read pulse width for one fixed size of disk drive and ignore the need for "write precompensation" altogether. This technique of simply omitting the adjusters is the crudest possible way to get over the need for adjustment, and it doesn't appeal to me; no amount of "works in practice" and "typically good error recovery" claims will convince me that the technique is satisfactory.

The need for the correct read pulse width is simple to explain; obviously, the chip should look for the data within a "window" which exactly covers the area where the expected data should be; the read pulse width is the width of that window.

Phenomenon

The need for write pulse compensation requires a more involved explanation, but very interesting nevertheless. It comes from the physics of magnetic recording that as flux transitions (areas of magnetic change) are packed closer together on the track of a disk, those spaced close together tend to move still closer together, those spaced apart move further apart and those in between stay where they are. The phenomenon only affects MFM (double density) recording because that has a range of differing spacings for flux transitions, and not (as we shall see later) because there is any decrease in the minimum spacing between adjacent flux transitions (there isn't).

Write enable precompensation takes notice of the fact that pulse positions will move in this way on the inner tracks of the disk effectively making their position "early" or "late", and compensates for this by recording the pulses deliberately "late" or "early" respectively, so that after they have moved magnetically, they end up in exactly the right place.

The 2797 chip has an output called TG43 (track greater than 43) which goes high when the head is positioned over tracks numbered greater than 43. This signal can be connected straight into the ENP (enable precompensation) input of the chip to apply the beneficial "write enable precompensation" at the inner tracks, We only hear of the need for this on 8" drives, but precompensation is equally desirable, for exactly the same reasons, on the inner tracks of other size drives, (and it costs nothing to apply it, because the feature is already built into the design of the 2797 chip for the purpose).

Piffle

The optimum place for precompensation to begin on 77 track 8" drives is at tracks greater than 43, and this is near enough the optimum place for 80 track smaller diameter drives. However the optimum for 40 track drives is around track 20, and the FDC-1 design allows the 2797 "ENP" input to be controlled additionally by the operating system to achieve the results which most nearly achieve perfection. I have recently seen a technical description of a well known supplier's floppy disk board which uses the 2797 family chips, which is foolish enough to state that precompensation is needed at the inner tracks of 80 track drives, but is unnecessary on the inner tracks of 40 track drives - what piffle! How does a disk drive head trundling round a particular inner track know how many steps it took to get to its present position? What these designers are saying is that they forgot to make arrangements for selectable precompensation at tracks other than track 43, and are trying to bluff their way out of it.

A further adjustment which is needed on the 2797 is the mid point setting of the "phase locked loop". The phase locked loop contains a voltage controlled oscillator which is locked to the frequency of the read signal coming from the diskette. If the diskette runs slower, so does the VCO (voltage controlled oscillator), if it runs faster, so does the VCO. With double density recording there aren't many clock pulses in the received signal, and the VCO fills in the gaps. The phase locked loop in the 2797 is obviously designed by experts (Western Digital should know what they're doing by now!) and can lock onto a signal very quickly; this is one of the reasons which enables us to minimise the space between sectors on our diskettes and fit 10 sectors on a track where others can only manage 8 or 9. Again I shan't go on, but I hope you can see that the phase locked loop and VCO is a very good idea, and well worth the penalty of needing an adjustment when first set up.

Phase Locked Loop

I have seen some competing designs, again from bigger firms than mine, which although wise enough to use the wonderful 2797, throw all the advantages away by sloppy implementation. For example, the phase locked loop has to have a "low pass filter"; we shan't go into detail here, save to observe that the capacitor in the filter obviously has to change for the two main groups of data transfer rates: 100nF for the faster group, and 200nF for the slower group. I have seen a fixed 100nF and the statement "works in practice for the other rate" (maybe it does, but thanks to the 2797, not to that designer!); another common arrangement is (ugh) to split the difference and use 150nF - and make everything equally wrong. Needless to say, I provide the two correct values, and suffer the expense (about 50p or so) of providing the chip to switch values electronically - would you thank anyone for saving you 50p by using incorrect component values in a critical circuit? You would? Then Interak is not the computer for you!

A further feature of the 2797 is that it has a "side select" output, that is to say the chip itself is in control of which side of the diskette is in use. This means that the two surfaces of a diskette can logically be amalgamated and appear to the computer system as one giant double sized surface. The 2797 writes the appropriate side number (0 or 1) as part of the identification field associated with each sector on the disk, and can immediately determine if an incorrect access is made. If the control of the sides was taken outside the disk controller chip, certain confusions could result. Once again I shan't go on but shall simply mention the ultimate absurdity which is the consequence of a naive designer not taking such matters into account, and that is the absurdity of treating a single diskette as being two separate half sized diskettes - one side could thus be full to overflowing whilst the other side was empty and unreachable. This is a total waste of the two heads in a double sided drive - effectively the same result could be obtained on a single headed drive by having a diskette which could be withdrawn and flipped over to access the other side, Amstrad fashion.

Inanities

I should of course not have been talking of the single and double density operation of the 2797 without saying something of what these terms mean. A disk recorded in double density does in truth store twice as much data per track as does one recorded in single density, and obviously works at double the data rate, since the diskette rotates no faster for double density operation. "Double Density" by this (correct) definition should not be confused with the home computer "double density" industry, which, too ashamed to own up to the fact they couldn't work in true double density instead pretend that doubling the number of tracks on the disk is double density. Then, when they eventually crack the secret of true double density they are driven to using such inane phrases as "true" double double density or "quad density". All is revealed when they have to quote their disk drive capacities: eg disk capacity 800K normal computer, 400K XYZ home (or school) computer. Further marketing tricks of this nature I have seen include that of adding the capacities of two disk drives together and quoting that, which again is double - you might as well go the whole hog and quote the capacity of a box or two of the floppy diskettes: 10 or 20 Megabytes! What is important is how many bytes of storage are available to one disk at one time.

Bright Spark

But back to the plot. The less emotive terms are "FM" (Frequency Modulation) for single density recording and "MFM" (Modified Frequency Modulation) for Double Density. FM is the easiest for me to explain (and I intend to quit before I have to explain MFM in too much depth!): An FM disk has clock pulses equally recorded throughout the disk. Imagine them like milestones along a road. Inbetween the milestones, ie at half mile increments, is a space where you look for data; a logic "1" could be represented by a half-mile stone, and a logic "0" by nothing at all. By this analogy a sequence of logic "1"s is represented by stones spaced at frequency of one stone every half-mile, and a stream of logic "0"s is represented by one stone every full mile - half the frequency. This is called frequency modulation: the higher frequency represents "1"s and the lower frequency "0"s. Easy to explain, and I hope, easy to understand. The method however does involve an awful wastage of stone; well over half the space on an FM diskette consists of milestones (or clock pulses, as we call them). Some very bright spark, at IBM I suppose, had enough genius to suggest that the milestones themselves could carry data: the data then would be spaced at half mile intervals, not one mile as before. The stones themselves would however be no closer together, and thus no harder to record, which explains why a single density disk drive will also record double density (and to an extent explains why single density diskettes are just as good as double density diskettes for recording data; the minimum distance between adjacent flux transitions is the same in both methods).

In MFM "1"s and "0"s are again represented by different frequencies, but they are quite scrambled compared to FM, and difficult to unravel. Unless you are given a very good start, of a sequence of some known data, it is very difficult to decode the "0"s and "1"s of an MFM recording. So difficult in fact that I wouldn't do it; I would simply pay my 19.90 and fit a 2797 instead (I said it was worth it). As most of the information content is now "data" and very little is now "clock" you can see the benefit in having a phase locked loop to deduce and maintain a steady clock, locked to the data rate. Silly me, I forgot to mention why you need a clock to recover the data: The clock in my analogy ticks away at a steady rate of one pulse every half mile, and it is only when the clock ticks that the 2797 looks at the road to see if anyone has left a stone there or not. You can see that the accuracy of the position of these stones is vital which explains why I was ranting and raving earlier about the importance of correct Write Precompensation and Read Enable Pulsewidth adjustments, because these enable the positions of those stones to be set and determined "just so".

This part of the lecture had better draw to a close. Suffice it to say that this design can cope with single or double density, (call it "quad density" if you like), and can do it properly.

Optimum

The 2797 can, if required, accommodate disks with over 250 tracks, so the present 35, 37, 40, 77, 80 track disks in common use are no problem. As mentioned to the point of boredom already, the 80 track 5.25" and 3.5" drives which step at 3ms can be stepped at this optimum rate in my design. (Exactly as you might have expected, but it shows how sloppy some designs are that I can boast about the fact that I offer the wonderful feature of simply being able to step at the correct rate.) And of course with suitable software different rates can be specified for other drives in a system.

If I were teaching you the alphabet I would start at

"Z" and work back to "A", sorry! I've just realised that I haven't really explained the general principles of recording data on floppy disks, so here we go: the floppy disk is so called because "disk" is the American spelling of our "disc": a circle. Wherever there is the chance of confusion between a "disk" meaning a disk drive, and a "disk" meaning the removable physical medium on which the data are recorded, I try to use the terms "drive" and "diskette" respectively; I don't always manage it though.

(My children are so computer literate that they would draw a square if you asked them to draw a disk, because, Daddy, disks are square. No my son, it is the disk jacket which is square, the round disk is inside.) The data is recorded on concentric circular tracks on each surface of a double sided diskette and on one surface of a single sided diskette. Each of those circles is a "track" and each track is divided into a number of "sectors".

Juggernaut

If a track were a line of cars going round a roundabout then each car would be a sector. Sectors can be small (Mini Metros) less small (Ford Sierras), just right (Luxury Coaches), or huge (Juggernaut Lorries), corresponding to 128 bytes per sector, 256, 512, 1024. You can see that according to this definition 512 bytes per sector is "just right", which is why I use that size myself. The tracks are numbered 0,1,2,3, etc starting from the outer track and moving inward, and the sectors are numbered 1,2,3,4 etc as you go round the roundabout. Note that tracks start at "0" and sectors start at "1". Offensive as this is to the logical mind, it would be a braver man than I who would alter such a well established convention, no matter how stupid it seems (and is!)

A few silly manufacturers have had a go at making a confusing system even worse, for example by numbering all the tracks conventionally on one side of the diskette and in reverse order on the other, starting sector numbering with "0", ie 0,1,2,3, using different sized sectors, even different densities on different tracks, different numbers of sectors on different tracks (by changing the speed of the diskette for different tracks), and one group of comedians (The Superbrain Computer designers, someone told me) chose to record all their "1"s as "0"s and vice versa! However, it is quite a legitimate ploy, for performance reasons, to scramble the order of the sectors on the track, why, I even do this myself (and call it "hard skewing"). There is only one standard format which is universally accepted, and that is IBM 3740 single sided single density format for 8" disks. (Another IBM standard, for double density, called IBM system 34 was later also defined, but that did not gain the same wide acceptance.)

Unfortunately it is only real men who use 8" diskettes now, the rest of us simply have to take our chances and put up with the fact that very rarely will a small diskette from one computer be readable in another computer. (Even IBM themselves have several different incompatible formats across their own range of personal computers - it shows the problems faced by the "compatible" manufacturers if IBM can't even make an IBM compatible!)

Cyclic Redundancy Check

Fortunately all of these formats are software selectable, in our design at least. Provided you are willing to write the software to operate it, our FDC-1 card can read or write any diskette anyone else can format, provided either they will tell you how they did it, or you can crack the code yourself. The 2797 and its phase locked loop can be of great help here because one of the commands it can obey is an instruction to read every scrap of information on a whole track. Not only the contents of the sectors, but all the spaces in between, the track identification bytes, the sector numbers, the CRC bytes; everything.

Oh, didn't I mention all these? The 512 (etc) byte sectors of course aren't just plonked bumper to bumper on the track. It would be impossible then for the disk controller to write fresh information on an individual sector without spoiling adjacent sectors. In fact it would be all but impossible for the controller even to identify any particular sector from all the others. To overcome these problems a good safety gap of defined data is written in between the sectors, and before each sector begins there is a long run of known data (so that the phase locked loop can lock on to the signal and develop a valid clock to help decode the data which follows) then some identification bytes. The full details are in the 2797 data sheet (see the end of this document for how to obtain this) for those who are interested, but I only want to mention that first there is a track number, then a side number (0 or 1), then a sector number, then a byte which says how long the sector will be, then a couple of CRC (cyclic redundancy check) bytes for this ID (identification) information, then some more space, then the sector data itself to be read or written and finally another couple of CRC bytes for the sector data.

Status Register

Data on a disk is read and written under the control of a program called the DOS or disk operating system (CP/M is a well known example of a disk operating system) and no competent DOS would dream of writing any sector data whatever until it had read all of the ID information and confirmed that it was as expected. For example, if CP/M is going to write sector 5 on track 7 it will first step the head inward or outward as appropriate until it lands on what it thinks is track 7, and will then read the next ID that comes along, and will report a "seek" error if any track number but the correct one is read. It then does a check that the sector number is correct before proceeding with the actual writing of the data on the sector. The way that the operating system satisfies itself that the ID information it reads is correct is by confirming that the CRC bytes are valid, and it does this by examining a particular register inside the 2797 called the "status register". The status register has the results of a test which the 2797 itself has carried out to confirm that the calculated CRC bytes match the ones which are recorded on the diskette itself. If the CRC bytes the 2797 put on the disk when the disk was written are not identical to the ones it calculates when it reads the disk then clearly something has gone wrong.

Polynomial

The CRC has to be calculated by the 2797 as it is written and read whilst the diskette continues to spin - the diskette can't be stopped while some slow old microprocessor does the calculation, it has to be there, ready. (In this and other respects the 2797 works much faster than a microprocessor possibly could, which is why I keep saying it is worth every penny of its purchase price.) The CRC is a the result of a glorious piece of mathematics in calculating a wierd polynomial (for the precise formula see the 2797 data sheet). The clever part of the mathematics is that the odds against a corrupted data record calculating out to give a correct CRC are very great indeed. If a more easily understood "checksum" was in use (the checksum is just like the total of the number of "1"s in a group of "0"s and "1"s) it is easy to see how an error which changed a "1" to a "0" would be unoticed if there was a similar error which changed a "0" to a "1" elsewhere. Not so with the CRC polynomial.

Various simple commands to the 2797 can result in the 2797 doing a lot of useful tasks; for example there is a single command which causes the 2797 to automatically seek and verify a track. A simple instruction, but one which involves a good deal of detailed work from the chip, which it is nice to have done automatically and efficiently.

In olden days diskettes used to be driven permanently by a mains powered motor, all the time the computer was on. The quality of engineering then, as now, was excellent and the drives were rated for years of continuous use without breakdown or maintenance. A solenoid (the "head load" solenoid) was provided so that

the drive head(s) and the diskette medium did not suffer excessive wear in consequence. Whenever a disk access was required the response was almost immediate: it only took a few tens of milliseconds for the head to be "loaded" onto the disk. Of course the disk controller chip was best advised not to read the diskette until the head had "settled". To suit drives (both mains and low voltage dc powered) which use a head load mechanism, the FDC-1 design provides an external monostable (of user selectable durations) which prevents the 2797 chip from using the signal from the head until it has loaded and settled.

Trend

A modern trend is to make drives as small and simple as possible. I don't think this is necessarily better, but it is the way of the world. A modern miniature drive typically has no head load mechanism; the head(s) load automatically as soon as the diskette is inserted. Unnecessary wear on the heads and diskettes is prevented by the simple expedient of turning the main drive motor off when the disk is not being accessed. (Early misgivings expressed by some old hands in the business about the wisdom of leaving two lumps of ferrous metal (the recording heads) parked on the minute magnetic domains on the diskette have proved to be unfounded. I don't like it myself, but I have resigned myself to the fact that it is perfectly normal nowadays - come back 8", all is forgiven!) It takes perhaps a second to run one of these new drives up to speed so the previous concerns about milliseconds of head load delay pale into insignificance when using modern drives which have on/off control of the motors.

Monostable

Of course if a one second delay were applied for each and every disk access, just to be on the safe side, in case the motors had been off, the speed of operation would be intolerably slow. (Don't think this is a hypothetical problem, many commercial computers do still force such unwarranted delays on their users, perhaps to convince them that they should pay the extra for hard disks which, like the old mains operated 8", rotate continuously and are therefore always ready for immediate use.) The problem is that unless the hardware is appropriately designed (as the FDC-1 is of course), there is no way the disk operating system can work out whether or not the diskettes are already turning, and it therefore has to play safe and wait. The FDC-1 design includes a retriggerable "motors on" monostable, which keeps the motors turning for several seconds after they have last been used, and furthermore buffers the control signal back onto the data bus so that before each disk access the operating system can determine if the drive motors are already running or not, and skip the start up delay if they are.

Proviso

I overstated my case above when I said that without a method such as used on the FDC-1 that the operating system has no way of determining whether or not the diskette is rotating or not. In many drives there is a signal which can be used: the "Ready" signal. The 2797 is organised so that it will make no accesses unless the Ready signal from the drives is valid. The assumption is that the drive only issues the ready signal when a diskette is mounted and rotating at the correct speed. The only catch here is that some drives don't have a ready signal, and some do but ignore the proviso about rotating at the correct speed. I have seen the absurdity in specifications of drives which have a "true ready" signal, and others with an "ordinary ready" signal, ie they issue the signal even if the drive is not ready, simply if there is a diskette mounted and merely rotating. Even this type of "ready" signal is of some use, for example the operating system can thereby test to see if a diskette it wants to write on is mounted, and if it isn't an appropriate rude message can be issued to the user. To suit this requirement (for a simple "ready" signal that is, not for a rude message), the FDC-1 has a retriggerable monostable used to develop a ready signal of this type; we call it an "artificial ready" signal. Index pulses from the diskette as it rotates continually trigger the monostable, which has a pulse width a little over the time for one revolution of the slowest diskettes; if there is no diskette, there are no index pulses, the monostable "times out" and the ready signal is removed. This is fairly crude, but it is better than the often used (but not by me!) alternative of simply strapping the ready line permanently to the ready state. In the FDC-1 design the artificial ready signal can be used in conjunction with the "motors on" monostable signal. Obviously if the motors are already on from a previous access, the diskettes (if mounted in the drives) will already be up to speed; the artificial ready signal then can be used with confidence that the drive is ready - the ready signal will not be present if there is no diskette mounted, which is the main information this signal has to impart.

Good Practice

It is good practice when the disk operating system detects an error for it to try again, because there are occasional good reasons why the data might have been missed; particles of dust, vibration, etc. Up to 10 "retries" are often recommended for this purpose.

Sometimes other designers rely on the error checking abilities of their disk controller chip to save them the effort of worrying about such subtleties as ready signals etc. After all, they argue, the disk must be rotating at the correct speed to get a valid seek, so it is safe to write on the track then. No it isn't! A decent controller chip (eg the 2797) can read successfully from a disk which is wildly below speed, and it would be very dangerous to perform a write operation on a disk in that state - the data would spread over much more than its allotted space. (Perhaps this is why some people can't fit the 10 sectors we can on a track?) Equally disastrous is that if the damage to the data is done at formatting time all subsequent use of that diskette is affected. Such a system may be (unknown to the user) permanently running on retries, for every sector which is read; this slows it down to a crawl because a whole disk revolution must be suffered for each retry, instead of several sectors being picked up on each revolution of the disk. You will appreciate that several revolutions per sector is dismally slow compared with several sectors per revolution!

It is easy to see how "dodgy" areas like this give rise to the rumours that such and such a manufacturer's disk drives are unreliable, or so and so's diskettes are substandard. Perhaps the "substandard" disks are just slightly thicker than the others and the disk drive comes up to speed a few milliseconds more slowly as a result. I have always been a collector of the contents of other people's rubbish bins. I have tried whole boxes of diskettes which some (other) computer users say are completely useless, and found them perfectly acceptable on Interak.

Indecent Haste

Some of these other computer users are so loyal to their own computer manufacturer that they will hear no criticism of him (who would like to admit they'd spent 399.99 pounds on a load of rubbish!) and argue that hundreds of thousands of users can't be wrong. Yes they can; as I said earlier it took me two years or more working on the FDC-1 card and listening to advice before I was sure it was right. If I'd been working to a production deadline the original board would have had to go out, rough edges and all. It is very laudable for manufacturers to meet published deadlines and delivery dates, but not if means they have to rush their products out with the design still not complete. With computers this is not usually a matter of life and death, but this indecent haste goes on in other fields where things are much more serious; I am thinking here about the disastrous launch of the American Space Shuttle for commercial reasons, even though the designers had misgivings about reliability. If a job as important as that can be botched for commercial reasons, it is easy

to see that well known computers can be equally botched in design.

Another form of data protection is built into the FDC-1 circuit, and that is the acceptance of the 2797 of the "write protect" input which most drives issue if the diskette is physically set to a "write protected" or "read only" state. This is done by sticking light proof labels (or removing them for some drives; confusing isn't it?) over notches cut into the diskette jacket, and interrupting beams of infra-red light. The modern 3.5" system is particularly neat because sticky labels are not used for this purpose; instead the diskette cartridge contains a little plastic slide which can be used and reused for this purpose as many times as you wish. If the write protect signal is present it is absolutely impossible for the 2797 to write on a correctly connected disk drive, no matter what commands to the contrary are issued. The system is not fool proof; a few fools of my acquaintance on being supplied with master files on write protected diskettes proceed to remove the write protection and corrupt the data on the diskette! Don't ask me why they do it but they do. The method only protects against accidental foolishness, but not deliberate foolishness; anyone can break anything if they're determined enough.

Vital

The 2797 takes all its timing from a master input clock of frequency 2.0 MHz for 8" diskettes and 1.0 MHz for minifloppies and the other smaller sizes. One of a group of similar chips in the 2797 family is the 2793 which makes this selection internally; I have chosen not to use the 2793 because that chip does not have the vital "side select output" for the efficient use of double sided diskettes. (There are a two other members of this family, but they have not been considered because they require an inverted data bus. Some alterations to the FDC data bus buffering could be made to accommodate these freaks but this was not done as the inverting variants are so rare. Supply problems with the 2797 are unlikely because it is "second sourced" by so many manufacturers, who anyway often don't bother manufacturing the inverting versions.)

The clock frequency can easily be done switched outside the chip but the side select information is best generated inside the chip for the reasons given a page or two ago when I was discussing that subject. in fact I use a 4.0 MHz Crystal oscillator and divide it either by 2 (to give 2.0 MHz) or by 4 (to give 1.0 MHz). 4.0 MHz was chosen because it is a small, inexpensive crystal, and dividing it by 2 or 4 guarantees an exact 50% duty cycle, which is what the manufactures of the 2797 implicitly specify at the highest clock input frequency. In some Interak systems the CPU runs at 4.0 MHz, but the temptation to use the system clock was resisted. Plenty of users may prefer to run with different system clock rates, and they will still want to be able to use their disks if they do. Also, coming up are breeds of "super Z80" enhanced Z80 chips which have clock rates 12 MHz and more.

Trap

Although a simple jumper link arrangement would allow the FDC-1 board to use the 8" data rates or the smaller ones, this would be useless for the exchange of data from one format to another - you can't pull the board out and alter a link whilst it is in use for transferring data! A simple electronic circuit for doing this switching is easy to devise . . . and easy to get wrong! Some poor designers (not I of course) fall into the trap of switching the clock frequency to the 2797 asynchonously, ie without reference to the present state of the clock input. It is very easy then to give a short pulse accidentally, which for an instant violates the minimum pulsewidth clock specification of the 2797. Inside the 2797 is a finely tuned microprocessor, running a "microprogram" (honestly!) and it can easily go wrong if one of its internal program steps is corrupted, even for an instant, by switching the clock frequency asynchronously whilst it is working.

The dreadful thing here for the unwary designer is that such a design error may not cause any trouble on test (the circuit can go right, as well as wrong), but will be a time-bomb ticking away until the board gets into production, then Ker-pow!, another set of unexplained random errors; maybe only once a week or once a year, but in this context even one error is one error too many. My method is to synchronise the switching of the clock frequency to the master clock itself; it is then impossible for a rogue short pulse to be generated in between times, thus making this circuit Kerpow-free.

The 2797 is laughingly described as a single chip floppy disk controller by its manufacturers. They obviously have forgotten all the other chips which are needed around it to interface to the system microprocessor and to the drive cable itself. However I plead guilty to a charge of using more chips than the absolute minimum possible. All of the 2797 signals in my design are buffered in and out, so that outputs can drive all reasonable loads adequately, and so that inputs present defined minimum loads on the external system. Most important of all, (at least it was in the early days when the 2797 was the most expensive chip you ever would purchase), is that the 2797 is safely cocooned from the outside world by layers of cheap 74LS and 74 series gates, buffers and drivers. This saves the worry of connecting directly to the 2797, (which it must be remembered is a static sensitive device), via long lengths of ribbon cable connected who knows where.

Tough

Even though they are ugly old chips, I have used tough 7406 or similar open collector drivers for the interface signals. They can easily sink the 33 milliamp current which flows through the 150 ohm pull-up resistor terminations fitted to many disk drives today. Very few LS chips can sink 33 mA, and those which can are rarely used, and difficult and expensive to obtain (another of my design rules is to use chips which I can get!)

There are two main standard floppy disk interface cable standards. One uses a 34-way cable, the other 50-way. The 50 way interface (used with good old 8" drives) is well standardised, but the 34-way interface has been "got at" by numerous tinkering manufacturers. The FDC-1 card suits every drive known (known to me that is; no doubt someone will produce yet another freak to confound me). All the signals which I know to be variable are routed to two plug in DIL (dual in line) Header areas where any unpleasantnesses can be taken care of. 8" allocations are very well defined, and there is little variation from manufacturer to manufacturer, 3.5" and 3" the same, but 5.25" is anybody's guess. (However even 5.25" drives are settling down now, and if produced recently will probably be very similar electrically to the standards for the 3.5" and 3" types).

I suggest that a 50 way header always be used regardless of whether a 50 way or a 34 way interface cable will be fitted (you can always fit a 34-way cable into a 50-way header, but 50-way into 34-way just won't go). Complete the DIL headers links to suit your drives; if you are running mixed 8" and smaller drives in the same system use the 34-way signal allocations, and scramble the signals by a "bump in the cable", or a simple adaptor board.

Thévenin

The use of 150 ohm termination resistors in many disk drives has already been mentioned. Similarly we have to provide termination resistors for the drive signals which are input to the FDC-1 card (I shan't go into this subject at length now, but take it from me, termination is required). I use a less burdensome variation of the 150 ohm termination in this design, the "Thévenin" equivalent circuit of 220 ohm pull ups and 330 ohm pull downs. To AC signals these are effectively in parallel, ie 150 ohms per line. Most 8", 5.25", and 3" drives can sink the current through the 150 ohm termination resistors, but some 3.5" drives (including the ones we supply) are very power and space conscious and do not provide the necessary drive chips to support such a load (I did not design them you see). They limit the

interface cable to 1 metre, and can only sink about 6 mA. Therefore the FDC-1 termination resistors have to be changed (I suggest 1k pull up and 1.5k pull down). I only mention all this so that I can crow about my foresight in providing most of the termination resistors in the form of plug in SIL (single in line) resistor packs, which can easily be changed. Some of my more independent users like to disregard my advice and they solder their resistor packs in; they will have to pull harder when they come to unplug them.

The FDC-1 card occupies 8 I/O Port Addresses. In the Interak system these are ports 80H ("H" means hexadecimal notation) to 87H inclusive. However the I/O Port starting address is not wired in but is selectable by a DIL Switch. This allows the FDC-1 Card to be used in non-Interak systems if need be, and also plugged as an additional card into an existing disk system at different I/O Port Addresses to add further disk drives, or more commonly to allow me to test and set up user's FDC-1 cards without losing my existing use of disks. (This is an excellent demonstration of the great wisdom of the Interak modular concept. You yourself might never need to plug more than one floppy disk interface card into an ordinary computer, but you can see here how useful it is to people like me to be able to do it.)

Provision

There are many areas where you might like to make some of your own decisions, and at places where various aspects of the design are a matter of personal choice I have made provision for push on Jumper Links to allow the user to follow his own wishes, or to suit his own particular drives.

An 18-pin 0.3"/0.6" DIL "patch area" is provided, where you can add a chip or selection switch, or extra components to suit some private modification you may wish to make to the design. This sort of activity is not encouraged, but if you must, the patch area will prove a great help. The FDC-1 has been designed with many features which may not be required in a specific application. What use is made of them depends on the disk operating system (CP/M in our case) and its implementation. For example, although the FDC-1 design allows for a choice of sector lengths of 128, 256, 512, 1024 bytes per sector, our implementation only deals with 2 of these (the others can be patched into the software if you must read or write that number). In essence the hardware has been designed to do anything, the software we use only does what is necessary, but if you can write software yourself you have total control over the FDC-1 card.

Facts

I am sorry to have gone on at such length but this has been a deliberate response to the often voiced comment that it is so hard for an ordinary person to make a decision on floppy disk matters because nobody will reveal the facts. The facts are before you, now it is up to you to judge what features you think a floppy disk interface should have.

CONTENTS OF KIT

The kit of components, which is sold separately from the p.c.b. itself includes 20+ resistors, 4 SIL resistors, 4 variable resistors, 20+ capacitors, 1 trimmer capacitor, 2 diodes, 1 4.0 MHz crystal, 24 integrated circuits, 2 DIL switches, 30 DIL sockets, 19 0.1" pitch pin assemblies, 2 DIL headers, and 16 JLinks.

ORDERING INFORMATION, PRICES

Description	Code	Price	VAT
Bare Board	BFDC1	19.50	15%
Manual	MFDC1HW	3.00	0%
Main Component Parts	PFDC1	62.09	15%
Optional Setting up Service	SUFDC1	25.00	15%
Chip Data Sheet	F2797	2.00	0%
Detailed Component Price list	FDC1/P	FOC	0%